Paper AAS 06-219

CASSINI-HUYGENS MANEUVER AUTOMATION FOR NAVIGATION

Troy Goodson, Amy Attiyah, Brent Buffington, Yungsun Hahn, Joan Pojman, Bob Stavert, Nathan Strange, Paul Stumpf, Sean Wagner, Peter Wolff, and Mau Wong

Jet Propulsion Laboratory
California Institute of Technology

16th AAS/AIAA Space Flight Mechanics Conference

Tampa, Florida January 22-26, 2006

AAS Publications Office, P.O. Box 28130, San Diego, CA 92198

AAS 06-219

CASSINI-HUYGENS MANEUVER AUTOMATION FOR NAVIGATION

T. D. Goodson, A. Attiyah, B. Buffington, Y. Hahn, J. Pojman, B. Stavert,
N. Strange, P. Stumpf, S. Wagner, P. Wolff, and M. Wong*

Abstract

Many times during the Cassini-Huygens mission to Saturn, propulsive maneuvers must be spaced so closely together that there isn't enough time or workforce to execute the maneuver-related software manually, one subsystem at a time. Automation is required. Automating the maneuver design process has involved close cooperation between teams. We present the contribution from the Navigation system. In scope, this includes trajectory propagation and search, generation of ephemerides, general tasks such as email notification and file transfer, and presentation materials. The software has been used to help understand maneuver optimization results, Huygens probe delivery statistics, and Saturn ring-plane crossing geometry.

The Maneuver Automation Software (MAS), developed for the Cassini-Huygens program enables frequent maneuvers by handling mundane tasks such as creation of deliverable files, file delivery, generation and transmission of email announcements, generation of presentation material and other supporting documentation. By hand, these tasks took up hours, if not days, of work for each maneuver. Automated, these tasks may be completed in under an hour.

During the cruise trajectory the spacing of maneuvers was such that development of a maneuver design could span about a month, involving several other processes in addition to that described, above. Often, about the last five days of this process covered the generation of a final design using an updated orbit-determination estimate. To support the tour trajectory, the orbit determination data cut-off of five days before the maneuver needed to be reduced to approximately one day and the whole maneuver development process needed to be reduced to less than a week.

INTRODUCTION

The Cassini-Huygens mission, an international effort to study the Saturnian system, was launched in 1997. The spacecraft's cruise trajectory ended with arrival at Saturn in 2004.

*Authors are members of the Cassini Navigation Team, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

That event marked the beginning of the tour trajectory, for which the prime mission ends in 2008. Both the sheer volume of trajectory corrections, numbering 162, and their frequency, some are as little as three days apart, necessitated some form of software automation for maneuver development. The maneuvers typically involve complex trajectory changes for gravitational-assist swingbys and complex design strategies including ΔV minimization.

The Cassini-Huygens program has developed the Maneuver Automation Software (MAS) specifically for this problem. MAS envelopes many individual tasks in one process. Relevant to this paper, it includes Navigation software for trajectory propagation and search, generation of ephemeris files, general automation of tasks such as email notification and file transfer, and production of presentation materials. The Navigation component of MAS is referred to as MOPS-MAS as it primarily automates Maneuver Operations Software (MOPS).

The term MAS is often used to describe the whole system and the part of the sytem maintained by the spacecraft office (SCO). In this paper, the name SCO-MAS [1] will be used for the latter; MAS for the former. This name also represents the near-equal status that component has with MOPS-MAS. SCO-MAS and MOPS-MAS essentially function as peers, processing data and passing files back and forth. Conceptually, SCO-MAS is seen as being on top of MOPS-MAS. Users typically interact with SCO-MAS directly; they don't usually initiate MOPS-MAS directly, SCO-MAS handles that transparently. This scope of this paper is restricted to MOPS-MAS.

The greatest significance of SCO-MAS and MOPS-MAS is in the broad use that the Cassini-Huygens mission has committed to. This sort of automation software could have been applied to a relatively small set of tasks. The Cassini-Huygens mission has committed to applying the automation software to essentially the entire maneuver design process, starting from the retrieval of Orbit Determination (OD) estimates and ending with the transmission of commands to the spacecraft. That's not say that this process isn't occasionally interrupted by meetings to coordinate use of the software, discuss the maneuver design, and approve the maneuver command sequence. (but presentation materials for these meetings *are* automated)

MOTIVATION

The motivation for the development of MAS has been the spacing of propulsive maneuvers in the Cassini-Huygens tour of the Saturnian system. It is worth noting that during the cruise trajectory, from Earth to Saturn, the spacing of maneuvers was such that development of a maneuver design could span about a month, involving several processes. Often, about the last five days of this process covered the generation of a final design using an updated OD estimate. To support the tour trajectory, the OD Data Cut-Off (DCO) of five days before the maneuver needed to be reduced to approximately one day and the whole maneuver development process needed to be reduced to less than a week.

Figure 1 shows the spacing, in days, of all the maneuvers during the Cassini-Huygens prime mission, spanning July 2004 to July 2008. Note, for example, that a large fraction of the maneuvers are each about 5 days apart. Also, there about 30 maneuvers in the first year of the prime mission and then there about 60 maneuvers between about August

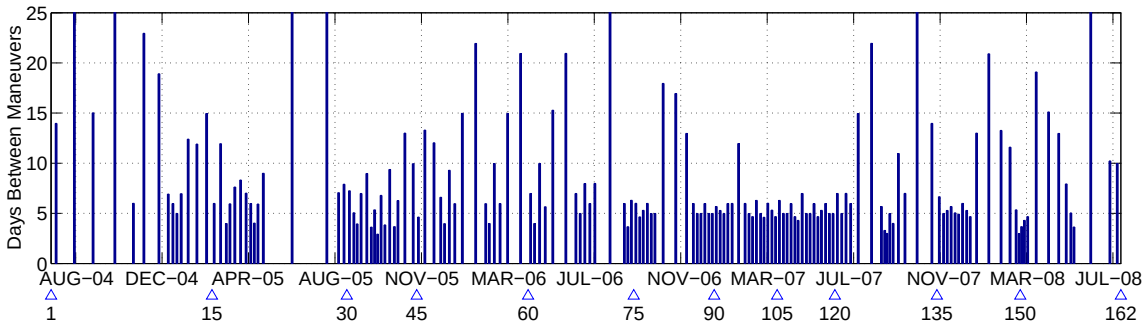


Figure 1 Maneuver Frequency. The seven clipped data points are 37, 46, 71, 26, 34, 27, and 43 days. The bottom row shows the cumulative count of maneuvers. Bars reflect time between maneuvers and so are placed midway between maneuvers.

2006 and August 2007. This creates a potentially stressful situation for the ground system. One way to visualize that stress is by placing events on a calendar, as seen in Figure 2. The calendar includes project-level maneuver-related meetings but excludes items such as periodic meetings for individual teams, internal team meetings per maneuver, and meetings for ongoing tasks like design of the extended mission. Looking at Figure 2, it is clear that there is little time to support mundane tasks like reviewing software output for problems or putting together a presentations, much less for doing analysis to understand the ΔV design, itself.

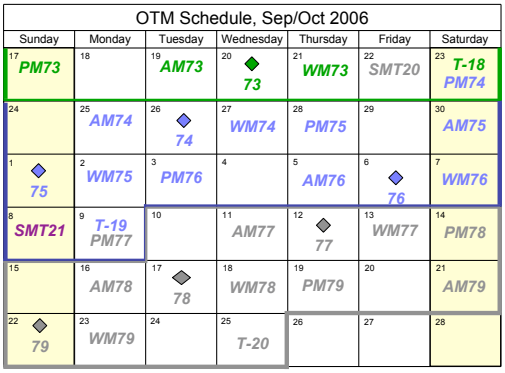


Figure 2 Events of Sep/Oct 2006. The left column is for Sunday. Diamonds mark maneuver execution and are above the maneuver number, text marks maneuver-related meetings, PM is preparation meeting, AM is approval meeting, and WM is wrapup meeting, SMT20 is the strategy meeting for the Titan-20 flyby, likewise for SMT21, and T-18, T-19 and T-20 mark flybys of Titan.

IMPLEMENTATION

MOPS-MAS is made up of components written primarity with Perl but including some Fortran and Matlab [2]. The Fortran software is mostly used to gather data from Fortran namelist input files. The Matlab software is used for numerical algorithms and for plotting results.

These components of MOPS-MAS are organized into two parts. The first part is called the MOPS-MAS Design software and handles the tasks necessary to generate a maneuver design and the deliverable files. The second part is called the MOPS-MAS Presentation software; it handles the gathering and generation of data about the maneuver design and the generation of presentation packages, reports, email messages, etc. used in meetings and project-wide delivery announcements.

This separation of the software is primarily so that different levels of configuration management may be applied to the two parts. The Design software requires very strict configuration management, "class A", as it is very influential in generating data for commands that are sent to the spacecraft for thruster firings. The Presentation software requires the opposite, a somewhat loose configuration management, because the type of data necessary for presentation to management may change fairly often and the consequences of errors in presentation material are generally mild. Loose configuration management allows for more frequent internal software revisions as needs for generating presentation material and reports are inherently more fluid.

The MOPS-MAS Design software interfaces with the Manuver Automation Software (MAS) of Cassini-Huygens' SCO team. That interface essentially provides for the delivery of files as depicted in Figure 3. The interface, a kind of Interprocess Communication (IPC), works within the computer's file system by reading, writing, and waiting for the existence of *flag* files. The software components that handle the IPC are referred to as the Requestor and the Listener. The data files also function as flag files; for example, the existence of the Maneuver Performance Data File (MAPDF) is used by the Requestor to signal the Listener that MOPS-MAS should be initiated.

The file exchange begins with the delivery from SCO to Nav of the Maneuver Performance Data file (MAPDF); it continues with the delivery from Nav to SCO of the Maneuver Profile file (MPF), ephemeris file, etc.; and, finally, ends with a hand-over from SCO to Nav of the Maneuver Implemenation file (MIF), MAS reports, etc. Generation of the MPF requires execution of the Navigation MOPS programs to generate a maneuver design. The MOPS-MAS Design software also generates ready-to-deliver OPTG, Light-time, and ephemeris files, among others. Further details are beyond the scope of this paper.

The MOPS-MAS Presentation software primarily gathers and computes report and presentation data from other files. The presentation software places its data into a simple database file, easily read and written with most programming tools, which is then used to fill in a number of templates. Formatting of data and some basic data manipulation is handled during the filling of templates. Some of these templates are simply text files which don't require further processing. Most templates do receive further processing, such as LaTeX input files, Matlab m-files, and executable scripts. The primary end-product is a ready-made

presentation (PDF file), including plots, for a maneuver-approval meeting. Other products include files to support real-time radiometric data monitoring, probability-of-impact data, report generation, file-delivery-announcement emails, maneuver-design-history tables, and more.

Pages from the presentation package for OTM-045 are reproduced in a fairly small form for Figure 9. OTM-045 used the single-maneuver, simple B-plane targeting strategy. The figure includes nine of the ten slides in the full package; omitted is the second slide which only delineated an agenda for the presentation. It is worth noting that these presentations are produced satisfactorily – they do not require any manual editing.

The presentation package in Figure 9 covers the following topics: comparison of the desired target parameters against OD's predictions of these paramter valuess. There are typically three plots of the B-plane showing both OD and maneuver-delivery statistics alongside the impact disk of the target body. A page of the pesentation is dedicated to data about the maneuver design , from spacecraft turn angles to achieve the burn attitude, to the day of the week that the maneuver will be executed. Also, with radiometric data being so fundamental to the OD process, the expected Doppler shift of the spacecraft's radio signal is plotted, observation of the real shift directly measures one dimension of the maenuver ΔV. Another slide shows the result of comparing two different computations of the spacecraft turn angles. The last slide lists information to help identify the run, including the user name of the person who executed MAS, a MAS ID for the run, and the time and date that the run started.

MANEUVER DESIGN

The MOPS-MAS Design software is essentially serving as a master script to handle a typical DPTRAJ/MOPS [3] runstream and to deliver output products. That runstream is basically the following sequence of software: MOPOUP, GINDRIVE, SEPV, TRNMPF, and TWIST. The program MOPUP reads data from the MAPDF, constructs input for SEPV, and for TRNMPF. The program GINDRIVE organizes inputs and options for DPTRAJ into a binary file called a Generalized Input (GIN) file. The GIN file is passed to SEPV and TWIST. SEPV solves for the maneuver ΔV. The program TRNMPF receives the ΔV design from SEPV and decomposes it into turn ΔV and burn ΔV; TRNMPF includes this data when it produces the Maneuver Profile File (MPF). TWIST prints a multitude of parameters, including orbital elements, describing the trajectory.

The MOPS-MAS Design software supports both single-maneuver B-plane targeting and a specific family of multiple-maneuver optimization strategies that use B-plane targeting. Some components do process data differently according to which maneuver strategy is being used. SEPV is a prime example because it actually implements the targeting strategy and computes the maneuver ΔV accordingly. Some other components, like TRNMPF and TWIST, are unaffected by the choice of maneuver strategy.

Single-Maneuver B-plane Targeting

For single-maneuver targeting, only the current maneuver is used to achieve the three target parameter values at the encounter body. This is used almost exclusively for B-plane targeting where the parameters are typically $\mathbf{B} \cdot \mathbf{R}$, $\mathbf{B} \cdot \mathbf{T}$, and time of flight. This type of problem is generally solved used a variant of the Newton-search algorithm. Figure 4 shows a B-plane plot, generated by MOPS-MAS for OTM-45, that depicts this kind of targeting. In that figure, the OD solution and post-maneuver delivery are drawn with their 1-σ dispersion ellipses. The arrow in the diagram has its tail at the OD solution and its head at the maneuver delivery, which matches the target. Also in Figure 4 is a corresponding plot of the time of flight, depicting the OD estimate and the maneuver delivery, the 1-σ dispersion is also indicated.

Multiple-Maneuver Optimization

Multiple-maneuver targeting is more complicated. The implemented optimization algorithm in SEPV was designed for the Cassini-Huygens tour; it constrains the combination of ΔV from the current and next maneuver to achieve the three B-plane target parameter values at the upcoming encounter body ($\mathbf{B} \cdot \mathbf{R}$, $\mathbf{B} \cdot \mathbf{T}$, time of flight). However, as each maneuver has three degrees of freedom, that problem is indeterminate. The remaining degrees of freedom are used to minimize the downstream ΔV for the next several encounters. Each of those downstream encounters may have a similar arrangement of maneuvers. [4, 5]

For maneuver-related meetings, it is often very useful to describe the results of the optimization problem in terms of B-plane targets, as depicted in Figure 5. When two maneuvers are combined in such a strategy, the target parameters are achieved by the combination of the two ΔV. Alternatively, one might saeuver has an optimized target, which is of little direct consequence, and the second maneuver achieves the desired target. Figure 5, produced by MOPS-MAS Presentation software, is an example of this arrangement; in that figure, the optimized aimpoint is referred to as "OTM-029 delivery" and the target aimpoint is referred to as "Final Aimpoint".

Other Targeting Strategies

MOPS-MAS supports other targeting strategies in the sense that it does not obstruct them. However, elements like the Matlab m-file for producing plots like Figures 4 and 5 are not flexible enough to make appropriate plots for the myriad of targeting options available in SEPV.

The maneuvers TCM-21*, OTM-001†, OTM-008‡ are examples of alternate targeting strategies. TCM-21 targeted a point on Saturn's ring-plane so that the spacecraft would pass cleanly through the gap separating the F and G rings on approach to Saturn [6].

*Any maneuver prior to Saturn Orbit Insertion (SOI) was referred to as a Trajectory Correction Maneuver (TCM) and any maneuver since SOI has been designed Orbit Trim Maneuver (OTM)

†Saturn-Orbit-Insertion Clean up

‡Huygens-Probe Targeting

OTM-001 used a critical-plane strategy in which the cronocentric orbit's semimajor axis and inclination were targeted and the remaining degree of freedom to the ΔV was used to minimize the magnitude of the ΔV. OTM-008 targeted the Huygens probe for entry conditions at Titan; those entry conditions were altitude, flight-path angle, and B-plange angle. The clean-up maneuver to OTM-008, OTM-009, used the same targets [5]. In each of these examples, specialized post-processing scripts were necessary to produce the appropriate plots and data for analysis and presentations.

CONFIGURATION FILE AND TEMPLATES

The design of MOPS-MAS has evolved during its development into a general structure for automated tasks, especially those that include report generation. This structure essentially relies on a simple, flat-namespace database and a scheme for filling-out templates.

All the high-level components of MOPS-MAS pass the database, called the configuration file, to each other. Figure 6 depicts this handling. The file is used for both input and output. It serves as the interface with the user and between each internal component of MOPS-MAS.

Any available option for any of these components may be set through a configuration-file key. The format of the configuration file is very simple; it is a plain text file. A line of the configuration file defines a key if it begins with a valid key name, followed by at least one space, and then followed by a string of characters, allowing for the hash character to mark the start of comments on that line. The term key refers to the name assigned to a piece of data, one may refer to a key as a variable name. This single file is the *primary* user interface. Figure 7 depicts a single line and denotes the key, value, and comment portions. The figure also shows the regular expression [7] that is used to pattern-match the line of text so as to identify the key and its value.

The configuration-file database serves as input and output file. It handles the input from one program to another program as well as the user's input. In the latter capacity, the file serves as an interface to the user.

Most components of MOPS-MAS are written in Perl. These components are generally wrappers that map configuration-file entries into program options, arguments, and map program output to configuration-file entries. For example, the configuration-file entry `SEPV_LOG` is a filename for the log output* of the DPTRAJ program SEPV. The key `TOTDVMAG_KMS` is the magnitude of the total ΔV as reported by SEPV. Furthermore, MOPS-MAS creates reasonable default values for many keys that are used for input to these programs, for example:

- The keys `START_TIME` and `END_TIME` hold calendar epochs for the beginning and ending of, among other things, ephemeris files. MOPS-MAS has algorithms, written in Perl, to compute these epochs based on the current date, the maneuver epoch, and the epoch of the targeted encounter.
- The key `MANEUVERNAME` holds a descriptive name for the maneuver being designed, for example *OTM-047*. It is derived from the more cryptic `MAS_ID` that is in the filename

*Unix `STDOUT`

Table 1
TEMPLATE EXAMPLE FILL_TEMPLATE READS DATA FROM THE CONFIGURATION FILE AND INSERTS IT INTO THE TEMPLATE TO PRODUCE THE OUTPUT.

Template	Configuration File

`The maneuver design for :MVRNAME based on the :ODNAME OD solution can be found in the following directory: :DELIV_DIR/:MVRNAMENODASH. This design yields a total delta-V magnitude for :MVRNAME of :TOTDVMAG(%.3f) m/s. :MVRNAME is to be executed :TSTART UTC SCET.`	`DELIV_DIR /nav/outputs`
	`MVRNAME OTM-047`
	`MVRNAMENODASH OTM047`
	`ODNAME 051228_020T10`
	`TOTDVMAG 1.827575112498210e-01`
	`TSTART 30-DEC-2005 02:47:00.000`

Output of `fill_template`	
The maneuver design for OTM-047 based on the 051228_020T10 OD solution can be found in the following directory: /nav/outputs/OTM047. This design yields a total delta-V magnitude for OTM-047 of 0.183 m/s. OTM-047 is to be executed 30-DEC-2005 02:47:00.000 UTC SCET.	

of the MAPDF, in this case `0047_a.mapdf`. `MANEUVERNAME` is used in the descriptive field of many different output files.

- The key `SAT6` holds a filename for the Saturnian satellite ephemeris from the OD solution. It is located by MOPS-MAS via a file-system search path input in the key `OD_INPUTS_ROOT`.

One goal of the MOPS-MAS software is to minimize the input required from the user for any given maneuver. A large part of this goal is achieved through keys like `START_TIME`, with computed default values. Allowing the users to set default values for any input key also goes a long way to achieving that goal.

The MOPS-MAS Presentation software uses the same configuration-file database that MOPS-MAS Design created. MOPS-MAS Presentation adds a wealth of data to the database and then leverages its the MOPS-MAS `fill_template` algorithm to create formatted output files from user-supplied templates.

The `fill_template` program reads data in the form of key-value pairs from the configuration file. `fill_template` scans an input template file for text formatted as a colon character followed by a key name, like `:MY_KEY`. It replaces the text `:MY_KEY` with the value of `MY_KEY`. The text `:MY_KEY` is called a template tag. See Figure 1 for an example.

Simplicity of this interface all but demands a flat namespace for the key names (variable names). For example, ephemeris files and other output files from a variety of programs are indexed by date and time. To keep consistency, all these files share a common timespan

denoted by the keys START_TIME and END_TIME; as such, these keys do not belong to any one particular program. Neither are these files all produced by a similar family of programs nor at quite the same point in the software. In other words, it proved difficult to conceive of any obvious logical grouping for these variables.

Unfortunately, there are hundreds of keys. Some are primarily for input and some are primarily for output. Many of the output keys are not used as input to other programs. These keys may be named according the software that produced them. This naming is not a formal grouping, only a naming convention.

Another reason to adopt such a naming convention is that some programs are double-checking other programs and so there is a need to store multiple computations of the same parameter. In order to avoid naming conflicts, grouping is necessary. For example, the magnitude of the ΔV from the Maneuver Implementation File (MIF) is named MIFBRN and the same data from the MPF is named MPFBRN.

Units are avoided here to maintain simplicity of the configuration-file database format and to ease programming requirements on post-processing scripts. The presentation software and the post-processing scripts are the least-critical elements as they do not provide products that are on the critical path for the maneuver's command sequence. Being least-critical, the concern of units should be lessened and tracking of units within the presentation software is not required. There is a clear drawback to this, but it is mitigated by committing to the documentation of units for all keys and by including units in the names of highly visible or easily confused keys, as in TOTDVMAG_KMS which is the total ΔV magnitude in km/s.

The example in Table 1 shows the sort of thing that may be done in plain text or for email messages. Hoewever, a template could be a rich-text-format (rtf) document to be opened and printed with a word-processing program. A template could be an input file for a mathematics package like Octave [8] or Matlab [2] that computes new values and generate graphics. A template could also be a LaTeXdocument, though which one can produce multi-page reports or presentation material including graphics.

Templates and the `fill_template` Program

A template file is any regular file, text or binary, that includes ASCII text data in the form of a template tag. `fill_template` replaces the tag text with the value of the key named in the tag. This, alone, is too simple; different reports have different levels of precision. Usually a presentation to management requires less precision than a summary of relevant data for an engineering team. Therefore, `fill_template` allows for a number of processing directives that may be added in parentheses, as in `:TOTDVMAG_KMS(%.3f)`. Here, the directive "%.3f" is taken as input to the Perl or C-language function `sprintf` [7]. It is distinguished from other directives by using the percent character as its first character. `sprintf` performs most conceivable number-formatting tasks.

The implementation of these directives via `fill_template` allows some additional processing to happen when templates are filled-out. More importantly, it puts the responsibility of formatting output into the hands of the users writing templates instead of the software

developers writing additional scripts or even users writing additional scripts. The reason this is so important is that the requirements on formatting data may change very rapidly for presentation material. A mere half-hour before a meeting, a manager may request that an engineer increase or decrease the number of digits being shown. These directives not only make it easy to quickly generate whatever formatting is desired, it keeps the information about formatting *in the template* where it is most relevant.

Directives may be strung together in a sort of chain using the `|` or "pipe" character*. By stringing directives together, more sophistication is possible. Based on the example in Table 1, examples of directives for `fill_template` follow:

- `:DELIV_DIR(b)` invokes the `basename` function which extracts the last element of the file path, in this case the text "outputs"
- `:TOTDVMAG(%.3f)` gets the result of `sprintf('%.3f',1.827575112498210e-01)`, which is 0.183
- `:TOTDVMAG(abs)` gets the absolute value of a numerical value, so `:TOTDVMAG(%.3f|abs)` and `:TOTDVMAG(abs|%.3f)` both give 0.183
- `:BRNDUR(%3.2f,10,%3.2e)` gets formatting like 6.48e-01 if BRNDUR is less than 10, e.g. `BRNDUR 6.47602768e-1`, or like 64.76 if BRNDUR is larger than or equal to 10, e.g. `BRNDUR 6.47602768e+1`. This is simply a conditional inequality layered over use of `sprintf`.†
- `:MVRTCA(ddTSTART|df%sign%dh/%hv:%mv:%sv)` gets a formatted date difference using the dates/epochs stored in keys TSTART and MVRTCA. Two directives have been piped in this example; `dd` computes a date difference in a standard, not-user-friendly format and `df` formats the result per user request. In this example, If MVRTCA contains `15-JAN-2006 11:42:31`, then the output will be `16/8:55:31` indicating 16 days, 8 hours, 55 minutes and 31 seconds between the two dates. Codes like `%hv` give value of the number of some measure between the two dates, but always positive; `%hv` gives hours, `%dv` gives days, etc. The sign of the difference is returned via `%sign`. When the sign is positive, `%sign` returns an empty string. The date difference, indicated by the characters "dd" at the beginnig of the directive, is computed by the function DateCalc and the result is formatted, indicated by characters "df", with the function Delta_Format, both from the Perl library Date::Manip [?].

Other directives for `fill_template` allow for addition of two keys' values, subtraction, multiplication, division, reformatting of date strings via the function UnixDate from the Perl library Date::Manip, and escaping special characters for LaTeX.

*The term "pipe" is borrowed from terminology of the Unix computer operating system

†The `sprintf` directive without the conditional *is* allowed to contain commas as long as it doesn't fit the pattern of `(#...,number,#...)`

Post-processing Scripts

The user may supply any number of post-processing scripts. Like the high-level components of MOPS-MAS, these must be fully-functional with only one input argument: the file-system path to the configuration file, as depicted in Figure 6.

Users may supply any post-processing script that they want. Within the Navigation team, these scripts are referred to as USER_EXEs because the configuration-file keys to specify them are `USER_EXE, USER_EXE_1, USER_EXE_2,` etc.* This has become a mechanism for handling maneuvers with special or unique targeting, adding specific computations to the automated runstream so that results may be used in presentations, testing bug fixes, and for adding generic features to MOPS-MAS Presentation.

An example of an added feature is the script called `BuildKey`. This script allows users to dynamically define new configuration-file key/value entries. Although more options are available, the most basic and essential feature is as follows. Any key named `BUILDKEY_{ID}_{VAR}` and whose value is like this "=expression" will cause `BuildKey` to create a new key named `{ID}_{VAR}` whose value comes from evaluating "expression" in Perl. Expressions via other languages and tools are possible via backticks, e.g. `BUILDKEY_MY_DATE =‘date‘` will create a new entry like "`MY_DATE Sat Jan 7 14:25:00 PST 2006`" and `BUILDKEY_A_HELLO =‘python -c ’print "Hello, world"’‘` will create a new entry "`A_HELLO Hello, world`". The significance of `BuildKey` is that a user need not write a whole post-processing script to add a new, minor feature. `BuildKey` allows users to enter one-line scripts in the configuration file and add features with very little effort. `BuildKey` was designed by a user not heavily involved in the development of MOPS-MAS.

An example of an extra, specific computation is a script named `Aimcheck` which determines the optimized target from SEPV's maneuver optimization. The information is not reported in SEPV's output and it requires a separate trajectory propagation, via DPTRAJ, to determine it. The DPTRAJ tracjectory print program, TWIST, is required to compute the B-plane parameters for this optimized aimpoint. That information is used produce B-plane plots as seen in Figures 5 and fig:simple-bplane.

USER'S RESPONSIBILITIES

The software saves hours of work and avoids many types of human error. However, core tasks of the job remain: set-up and analysis. A significant deviation from a manual process is that an automated process requires user's to get the set-up ready far in advance of operation.

Users are responsible for setting up MOPS-MAS in general, across all maneuvers, and for each particular maneuver. This job is large; it includes many settings and several files. Perhaps the largest of these files is the MOPS-MAS default configuration file. This is the configuration file loaded first and contains nearly all the default settings for MOPS-MAS. For example, `MVR_INPUTS_ROOT` is the file path within which MOPS-MAS will look for other input files for the maneuverdesign. `OD_INPUTS_ROOT` is the file path within which MOPS-

*The scripts are executed in respecive order, `USER_EXE` is first, `USER_EXE_1` is second, etc.

MAS will look for files that comprise the latest, official OD estimate for the maneuver in question.

An example of a set-up task is the choice of B-plane plots to include in presentation material. MOPS-MAS produces a standard palette of plots for the user to select from. These are actually all the same plot, but each case has the plot limits set to highlight a different set of features. An example palette is shown in Figure 8. "od+imp" only tries to include the OD ellipse and the center of the impact disk, "mvr+final-zoom" attempts to zoom-in on the an area that only contains the target aimpoint ("mvr") and the final aimpoint. Generally, it seems that a satisfactory choice, in general, is as follows: "od", "all", then "mvr".

On a per-maneuver basis, users are responsible for MOPS-MAS configuration in addition to the usual items. Even without MAS, users would have to edit Fortran namelist input files for DPTRAJ and some utilities. Navigation must edit such input files to ensure use of the appropriate B-plane targets and targeting strategy for the maneuver.

As part of the set-up task, users are also responsible for maintaining and updating the template files. There are many template files, some more important than others. Contents of these files range from items like descriptive comments concerning the maneuver design, assessment of the OD estimate versus the trajectory target parameters, email text for internal and external (relative to the Navigation team) delivery announcements, data to assist in analysis of the maneuver design, and a variety of presentation materials.

SPECIAL CASES

Two special cases of maneuver design and analysis further illustrate both why maneuver automation has been necessary for Cassini-Huygens and its fringe benefits.

Ring-Plane Crossing

On approach to Saturn, the spacecraft would pass through the gap separating the F and G rings. A handful of debris fields were identified as keep-out zones for this trajectory. Maneuver TCM-21 targeted the ring-plane crossing point and was relied upon to produce a trajectory that kept out of the keep-out zones in order to minimize the probability of damaging the spacecraft.

In support of this effort, a substantial post-processing script was developed to plot visualizations of the trajectory and uncertainties (OD uncertainty and maneuver delivery dispersions) to help assess the expected performance of TCM-21. However, TCM-21 did not have an unreasonably short development schedule.

Instead, a short schedule was the hallmark of the contingency ring-plane targeting maneuver, TCM-22. TCM-22 was scheduled at the last maneuver opportunity, just several days before pericrone. The maneuver would only be executed if, among other scenarios, it was determined that TCM-21 did not perform well or if TCM-21 was executed at all.

The same post-processing scripts used for TCM-21 were compatible with TCM-22, meaning that the Navigation had full confidence as TCM-21 was being designed and analyzed that TCM-22 would be prepared for. This is not to say that the maneuver automation software replaced test-and-training exercises of the ground system, for it did not; instead, MAS provided automation of software runs that would help evaluate whether or not TCM-22 was necessary.

Huygens Probe Delivery

As was true for other probe-carrying missions like Galileo and many missions to Mars, there is no second chance or correction opportunity once a probe is released *. Therefore, teams expend great effort in the time leading up to the separation event to gain confidence for success of the mission.

The Navigation team proved to be no exception to this for the Huygens probe delivery. The Huygens probe was released on December 24, 2004 following more than a week of daily navigation analysis. The OD team produced daily estimates of the spacecraft trajectory. The maneuver team produced, based on the OD estimates, predictions of probe-entry statistics for the entry-angle and the angle-of-attack. These statistics were compared to earlier studies to provide confidence that the Navigation system was performing as expected.

Post-processing scripts written specifically for Navigation of the Huygens probe mission were run as part of every maneuver design. The scripts included the aforementioned statistics for the official design and an alternate design strategy. The presentation package produced by MOPS-MAS was seventeen pages long and captured important data about both maneuver design options. Every day, the updated results were presented to the whole Navigation team. In this way, Navigation had confidence that on any given day both strategies were understood and performance of the Navigation system was being thoroughly tracked.

LESSONS LEARNED

The development and use of software to comprehensively automate ground system tasks is an emerging practice. From the experience of the Cassini-Huygens mission so far, there seem to be are observations and lessons learned that may be relevant to other efforts.

Perhaps the greatest of these has been the pitfall of underestimating the effort. Roughly speaking, the task of developing and putting MAS into use was originally expected to be a two-year process. Although it's difficult to measure how long this really took, it seems more accurate to say that it look at least three years and may have been closer to four years.

In some ways, the automation of a process is only as strong as its weakest link. If one team fails to automate their processes to an extent that matches the other teams, then the additional time that team requires to complete their work will slow down every team that is waiting on their products. So, slowing down part of a mostly linear process means that all downstream processes are affected and potentially slowed.

*Missions like Deep Impact are exceptions to this generalization.

There seems to be certain aspects of organizational culture that work against some of the ideas of automation. For example, some users might say "I don't feel comfortable delivering products when I don't know what was run". A second is: "I don't want someone else running my software". Part of the issue in both cases is that users seem to have difficulty with the idea that the owner of the output products is the person or group who *configured* or set-up the run, not the person or group who executed the run.

It is more difficult for users to learn what the elements of the maneuver-design process are when those elements are automated. Without automation, the user has to perform each step manually and, therefore, learns what those steps are. When the steps are all automated, it is more difficult to find the motivation to learn them because that knowledge *appears* to be unnecessary; it is also more difficult to discover what all those steps are because it is difficult to produce all the necessary documentation.

CLOSING

Experience with the Cassini-Huygens spacecraft has been very successful and should help enable exciting science investigations of the Saturn planetary system.

About 50 of the 162 maneuvers have been executed so far; all of them processed via MAS. This automation has been meeting its requirements and has reduced the workload during operations. The MOPS-MAS software has provided a favorable trade for the Navigation team between complexity in software versus time, workforce, and reliability. Another indication of success for the architecture of MOPS-MAS is that, as operations have progressed, the Navigation team members have been embracing the software to automate more processes.

APPENDIX: B-PLANE DESCRIPTION

Planet or satellite targeting is described in aiming plane coordinates referred to as *B-plane* coordinates[9] (Figure 10). The B-plane is a plane passing through the body center and perpendicular to the asymptote of the incoming trajectory (assuming 2 body conic motion). The "B-vector", $\mathbf{B}$, is a vector in that plane, from body center to the piercing-point of the trajectory asymptote. The B-vector specifies where the point of closest approach would be if the body had no mass and did not deflect the flight path. Coordinates are defined along three orthogonal unit vectors, $\mathbf{S}$, $\mathbf{T}$, and $\mathbf{R}$ with the system origin at the body center. The $\mathbf{S}$ vector is parallel to the spacecraft V_∞vector (approximately the velocity vector at the time of entry into the gravitational sphere of influence). $\mathbf{T}$ is parallel to a convenient reference plane, and $\mathbf{R}$ completes an orthogonal triad with $\mathbf{S}$ and $\mathbf{T}$. The reference plane for the $\mathbf{T}$ vector is generally the ecliptic plane (EMO2000). For Titan equator of date, the reference plane is in Titan's equatorial plane at the given epoch. With $\mathbf{S}$, $\mathbf{T}$, and $\mathbf{R}$ thus defined, a target point can be described in terms of the B-vector dotted into the $\mathbf{R}$ and $\mathbf{T}$ vectors ($\mathbf{B} \cdot \mathbf{R}$ and $\mathbf{B} \cdot \mathbf{T}$), or as the magnitude of $\mathbf{B}$ and the angle ϕ clockwise from $\mathbf{T}$ to $\mathbf{B}$.

Trajectory errors in the B-plane are often characterized by a one-σ dispersion ellipse, shown in Figure 10. SMAA and SMIA denote the semi-major and semi-minor axes of the ellipse; θ is the angle measured clockwise from the T axis to SMAA. The dispersion normal to the B-plane is typically given as a one-σ time-of-flight error, where time-of-flight specifies what the time to swingby (periapsis) would be from some given epoch if the magnitude of the B-vector were zero. Alternatively, this dispersion is sometimes given as a one-σ distance error along the **S** direction, numerically equal to the time-of-flight error multiplied by the magnitude of the V_∞ vector.

ACKNOWLEDGMENT

The work described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

1. *Maneuver Automation Software (MAS) User's Guide*. JPL Internal Document D-28416.
2. Cleve Moler. *Numerical Computing with MATLAB*. Society for Industrial and Applied Mathematics, Jun 1994.
3. *DPTRAJ-ODP User's Reference Manual, Vol. 1*. JPL Internal Document 630-336.
4. Yungsun Hahn. A new baseline maneuver strategy for Cassini T18-5 tour. *JPL IOM 312.H-01-005 (Internal Document)*, March 2001.
5. Sean V. Wagner, Brent B. Buffington, Troy D. Goodson, Yungsun Hahn, Nathan J. Strange, and Mau C. Wong. Cassini-Huygens Maneuver Experience: First Year of Saturn Tour. *Proceedings of the AAS/AIAA Astrodynamics Specialists Conference, AAS 05-287*, Aug 2005.
6. Troy D. Goodson, Brent B. Buffington, Yungsun Hahn, Nathan J. Strange, Sean V. Wagner, and Mau C. Wong. Cassini-Huygens Maneuver Experience: Cruise and Arrival at Saturn. *Proceedings of the AAS/AIAA Astrodynamics Specialists Conference, AAS 05-286*, Aug 2005.
7. Larry Wall, Tom Christiansen, and Jon Orwant. *Programming Perl*. O'Reilly, 3rd edition, Jul 2000.
8. John W. Eaton. Octave: Past, preset, and future. In K. Hornik and F. Leisch, editors, *DSC 2001 Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, Vienna, Austria, Mar 2001.
9. W. Kizner. A Method of Describing Miss Distances for Lunar and Interplanetary Trajectories. *JPL External Publication 674*, August 1959.

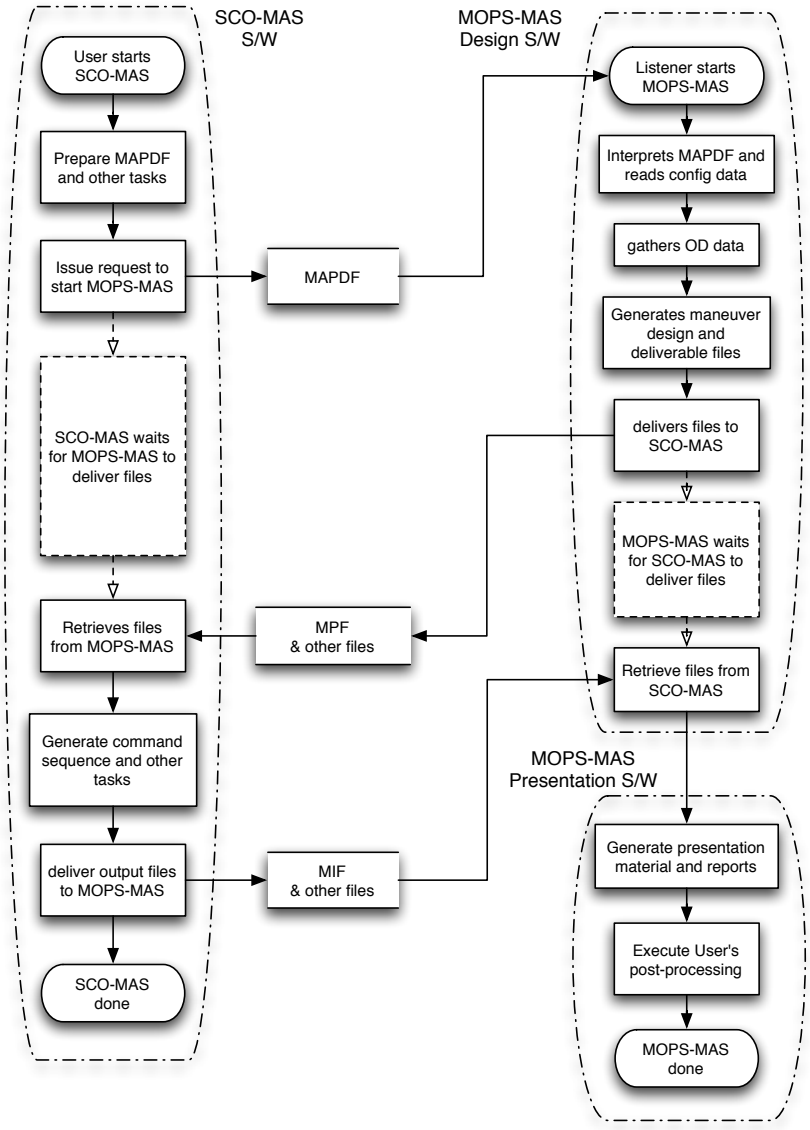


Figure 3 MAS Diagram. Shows the basic elements of interaction between SCO-MAS and MOPS-MAS

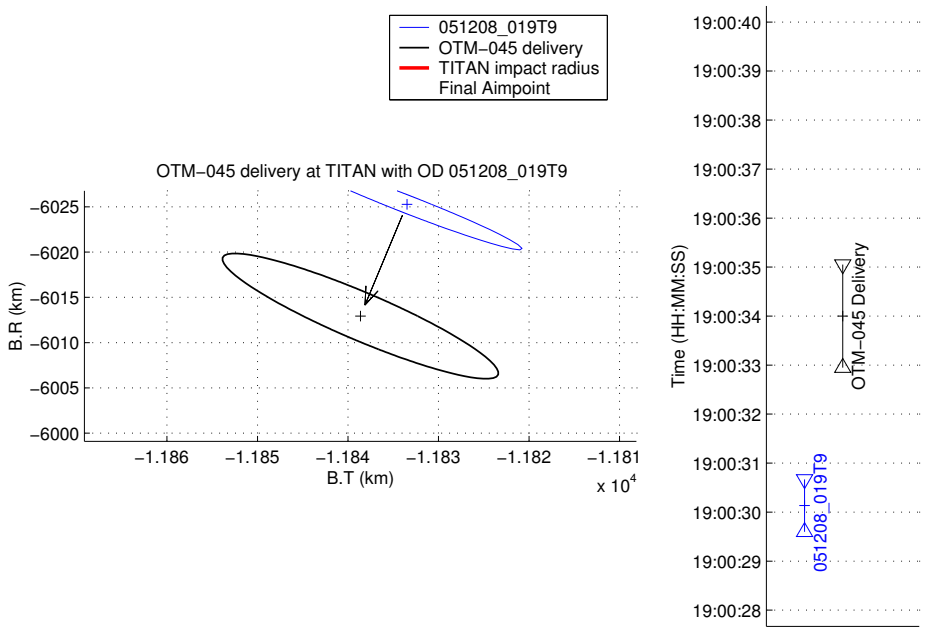


Figure 4 Automated B-plane Plot Shows aimpoint sequence for single-maneuver targeting. Plot on left side is B-plane and plot on right side is for time of flight. Both show estimates and 1-σ uncertainties

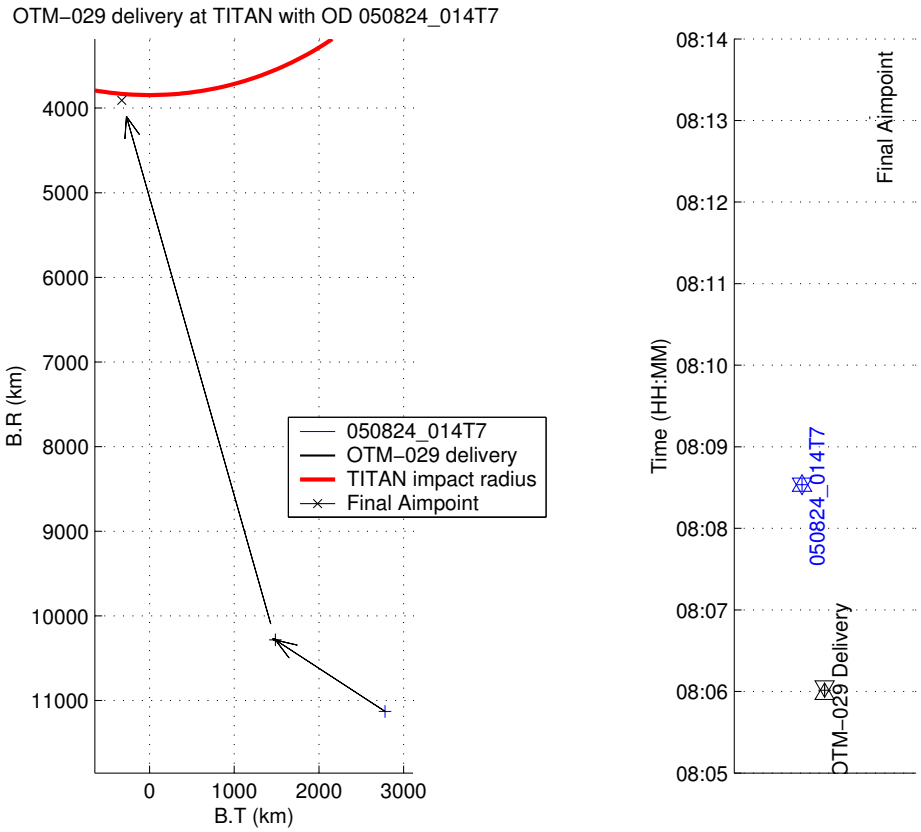


Figure 5 Automated B-plane Plot Shows aimpoint sequence for multiple-maneuver targeting. The red arc is part of the impact disk of Titan. The plotted aimpoints, starting from the impact disk and moving out, are: the final aimpoint, the intermediate/optimized aimpoint, the OD estimate. The intermediate aimpoint is also the expected delivery aimpoint from the current maneuver. See also Figure 4

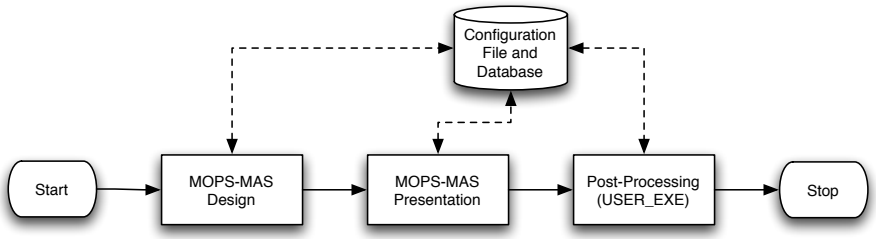
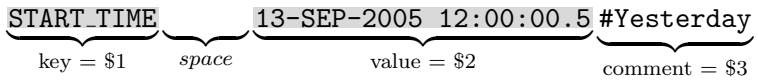
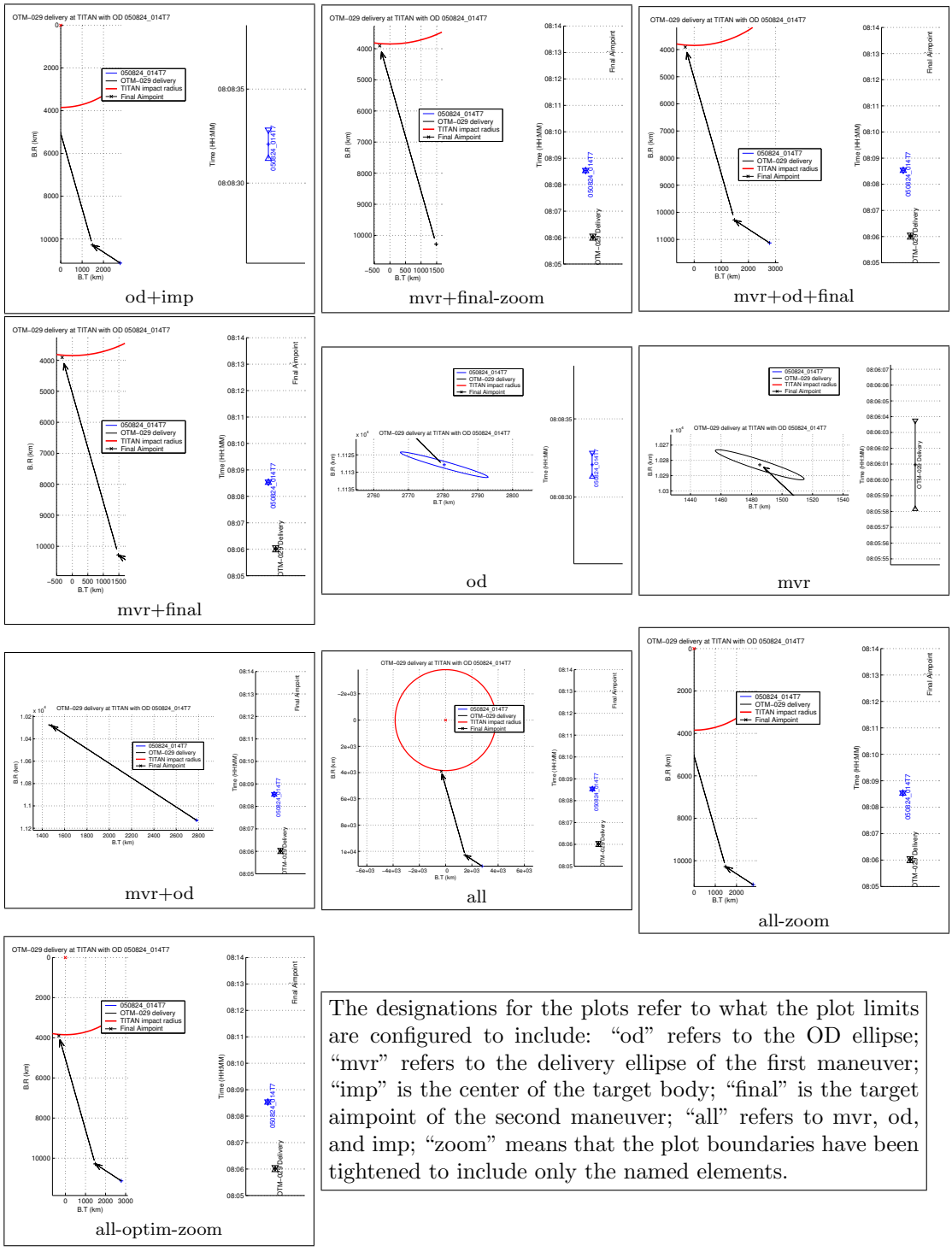


Figure 6 MOPS-MAS Configuration File. The configuration-file database is passed to and from each component of MOPS-MAS as input and output.



Perl regular expression: `^([A-Z0-9_]+)\s+([^#]+)((#.*)?)$`

Figure 7 Configuration File Entry. Any given line of the configuration file must match the given regular expression for the data to be accepted. The 'key' and 'value' are determined by so-called capturing parantheses, comments are not stored.



The designations for the plots refer to what the plot limits are configured to include: "od" refers to the OD ellipse; "mvr" refers to the delivery ellipse of the first maneuver; "imp" is the center of the target body; "final" is the target aimpoint of the second maneuver; "all" refers to mvr, od, and imp; "zoom" means that the plot boundaries have been tightened to include only the named elements.

Figure 8 Palette of plots automatically produced. Users may choose which plots to include in the presentation material. ' The vertical plot to the right-hand side reflects time-of-closest-approach data.

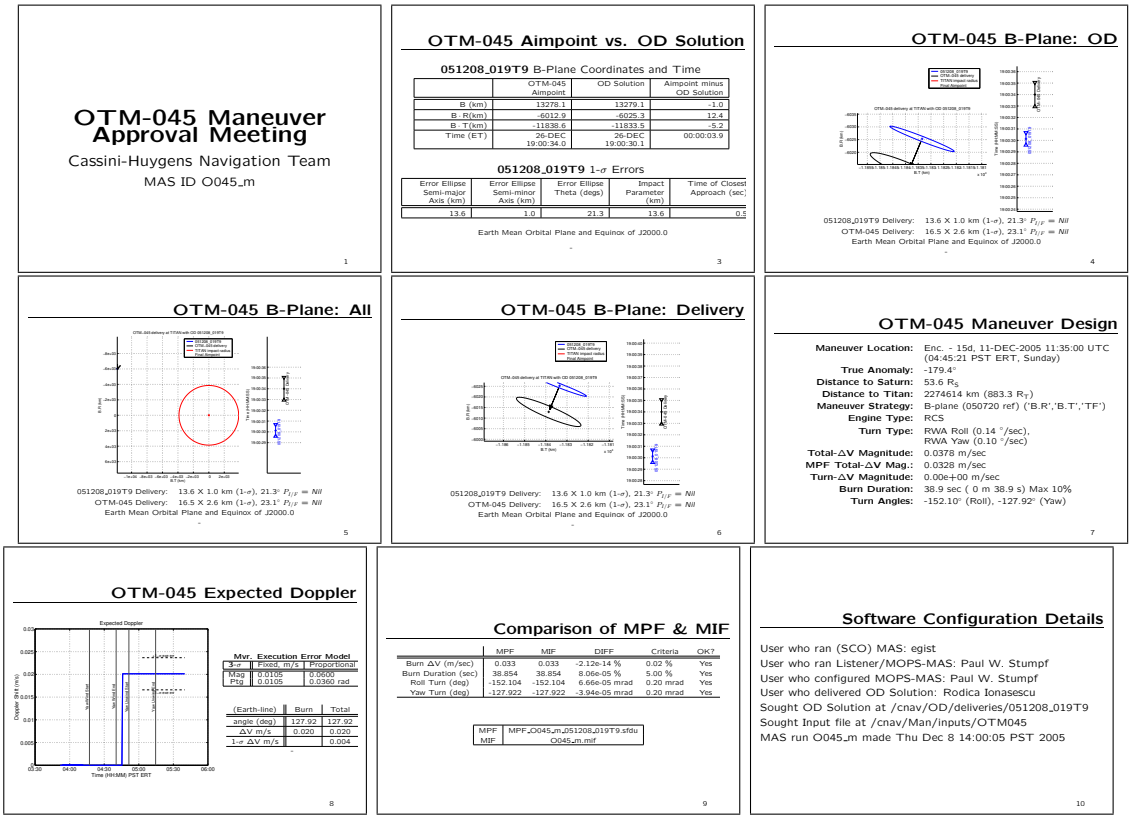


Figure 9 Pages from the OTM-045 Maneuver-Approval Presentation Package.

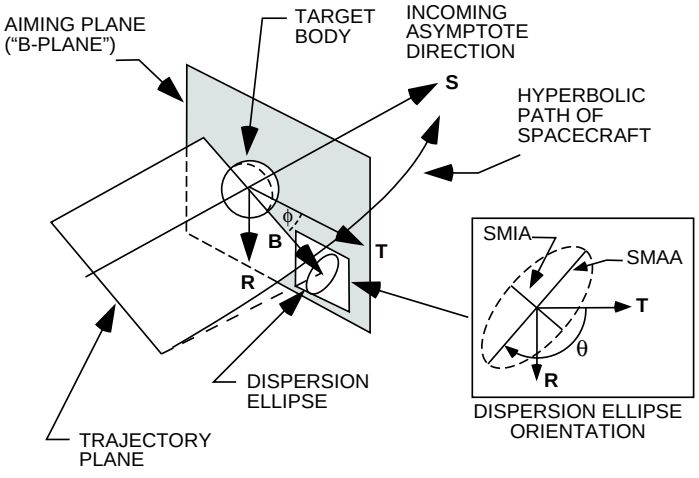


Figure 10 B-Plane Coordinate System.